Student Name: Student id: Serial #:

QUESTION #	1	2	3	4	5	TOTAL
MAX POINTS	6	12	10	8	10	
POINTS EARNED						

University of Bahrain College of Information Technology
Department of Computer Science

ITCS332: Concepts of Programming Languages SECOND TEST_EV **Date: DEC 13 , 2005**
****

QUESTION ONE: TRUE / FALSE Questions [6 pts]

1) ___F___ C++ allows nested subprograms. (Java & PHP)
2) ___F___ Dynamic Type Binding improves the ability of compilers to detect errors.
3) ___F___ Short names are more readable and less writable than long ones.
4) ___T___ Case sensitivity improves reliability.
5) ___F___ In FORTRAN, the special word "REAL" is a reserved word.
6) ___T___ Variables of 2 derived types are incompatible.
7) ___T___ The L-value means the address of the variable.
8) ___T___ Javascript is an example of languages using dynamic type binding.
9) ___T___ Type inferencing means types are determined from the context of the reference.
10) ___F___ A binding is static if it occurs during run time and remains unchanged throughout program execution.
11) ___F___ Languages with great deal of coercion such as C, are more reliable than those with little coercion such as Ada.
12) ___T___ In C++, "typedef" does not introduce a new type, it simply defines a new name for an existing type

- In static - Scope L.e. local variable plus all of visible var. in all of enclosing scope.
- ~ dynamic

QUESTION TWO: **Fill in blanks Questions** **[12 pts]**

- In C++, the variables allocated with new operator and deallocated with delete operator are of EXPLICIT-HEAP DYNAMIC type.
- Dynamic Scope is based on CALLING SEQUENCES of program units, not their spatial textual layout.
- Name 2 strings length design options: STATIC LENGTH STRINGS and T LIMITED LENGTH DYNAMIC STRINGS,
- The 2 types of type compatibility are: NAME COMPATIBILITY and STRUCTURE COMPATIBILITY
- The advantage of implicit declaration is BETTER WRITEABILITY and the disadvantage is LESS READABILITY AND LESS RELIABILITY.
- According to the storage binding, scalar variables are classified into 4 categories: static, stack-dynamic, EXPLICIT-HEAP DYNAMIC, IMPLICIT-HEAP DYNAMIC.
- The two problems associated with pointers are: DANGLING POINTERS and MEMORY LEAKAGE and DEREFERENCING UNINITIALIZED POINTERS.
- **Storage bindings & Lifetime of variables:**

  Study the following C++ f() function and answer the 5 questions below:

```
int *f ()
{
    int *p = new int(88);
    static float d = 3.14;
    return p;
}
```

  - The type of a pointer variable p is STACK DYNAMIC.
  - The type of the object (variable) pointed to by p is EXPLICIT-HEAP DYNAMIC.
  - The lifetime of a variable p begin when THE FUNCTION F IS INVOKED and ends when THE FUNCTION F IS EXITED.
  - The lifetime of a variable d begin when THE PROGRAM IS LOADED and ends when THE PROGRAM IS EXITED.
  - This code suffers from a problem of MEMORY LEAKAGE.

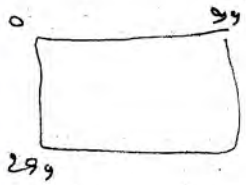
QUESTION THREE: **ARRAY TYPES** **[10 pts]**

- Given a matrix `F: array [0 .. 99, 0 .. 299] of float;` located at memory address starting at 2500. Element size is 4 bytes.

  a) Assuming row major ordering, calculate the address of matrix element F[80, 100].

```
F[80,100] = 2500 + 4 * ( (80-0)*300+(100-0))
          = 2500 + 4 * (80*300+100)
          = 2500 + 4 * (24000+100)
          = 2500 + 4 * 24100 = 966500.
```

  b) Assuming column major ordering, calculate the address of matrix element F[75, 120].



```
F[80,100] = 2500 + 4 * ( (120-0)*100+(75-0))
          = 2500 + 4 * (120*100+75)
          = 2500 + 4 * (12000+75)
          = 2500 + 4 * 12075 = 50800.
```

- Give the access formula for a 2-dimensional array `T[I, J]` with I index: IL: IU, J index JL:JU, and base BASE, element size ESIZE. Assume you are accessing element located at I , J for `column major` ordering. In other words, show how to compute the address of element `T[I,j]` for an array declared like this:

  `T: array [IL..IU,JL..JU] of element_type;`

  where T is stored in memory at address BASE and `sizeof(element_type)` is `ESIZE`. Show how the expression can be rewritten to save runtime computation.

  Let `m` be the number of elements in a column, so `m = JU - JL + 1`.

```
T[I, J] = BASE + ESIZE * ( ( (J) - JL)*M + (I - IL))
        = BASE + ESIZE * (M*J - M*JL + I - IL)
        = BASE + ESIZE * (-IL - M*JL + M*J + I)
        = BASE - ESIZE * (IL + M*JL) + ESIZE * (I + M*J)
```

QUESTION FOUR: Consider the following Pascal skeletal program and assuming the program begins execution from procedure main, answer the 2 questions below: **[8 pts]**

```
Program main;
var x: integer          {This is main:x}

procedure sub1;
  var x: integer {This is sub1:x}
  begin
    x := 1;
    sub2;
  end;

procedure sub2;
  procedure sub3;
    var x: integer; {This is sub3:x}
    begin {sub3}
      x := 3;
    end; {sub3}

  begin {sub2}
    x := 2;
    sub3;
  end;      {sub2}

begin {main}
  x := 0;
  sub1;
end. {main}
```

SUB1:X		SUB1:X
SUB3:X		SUB3:X
MAIN:X		SUB1:X
MAIN:X		MAIN:X

a) Notice the three variables called "x" have comments labeling them "main:x", "sub1:x", and "sub3:x".
Assume **static scoping**: For each assignment statement in the program, write next to the assignment statement either "static = main:x", "static = sub1:x", or "static = sub3:x" indicating to which variable named "x" the assignment statement refers when static scoping is used.

b) Repeat part a) for **dynamic scoping**. That is, write either "dynamic = main:x", "dynamic = sub1:x", or "dynamic = sub3:x" next to each assignment statement.

QUESTION FIVE: **Consider the following pseudo-code:** **[10 points]**

```
void sub1()
{
  int a, b;
  ... <---------------------- 1
}                /* end of sub1   */
void sub2 ()
{
  int b, c;
  ... <---------------------- 2
  sub1;
}                /* end of sub2*/
void main ()
{
  int c, d;
  ... <---------------------- 3
  sub2 ( );
}                /* end of main*/
```

- Assuming a static-scoped language, the referencing environment at point 1 is:

  a and b of SUB1

- Assuming a dynamic-scoped language, the referencing environment at point 1 is:

  a and b of SUB1 **c of SUB2** **d of MAIN**

- Assuming a static -scoped language, the referencing environment at point 2 is:

  b and c of SUB2

- Assuming a static -scoped language, the referencing environment at point 3 is:

  c and d of MAIN

- Assuming a dynamic-scoped language, the referencing environment at point 3 is:

  c and d of MAIN

Student Name: **Student id:** **Serial #:**

QUESTION #	1	2	3	4	TOTAL
MAX POINTS	10	10	11	12	
POINTS EARNED					

University of Bahrain College of Information Technology

Department of Computer Science

ITCS332: Concepts of Programming Languages **SECOND TEST** **Date: MAY 16, 2006**

****

QUESTION ONE: **MCQ Questions** **[10 pts]**

1) The activity of ensuring that the operands of an operator are of compatible types is called _____.
   a) Type checking b) Type binding c) Operator binding d) Address binding
2) The application of an operator to an operand of an invalid type causes _____ error.
   a) Address b) Lexical c) Scope d) Type
3) A garbage collection strategy that keeps track of how many pointers point to a particular block and puts memory back on the free-list when the count reaches zero is called:
   a) Reference counters b) Event counters c) Lock and key d) Tombstone
4) In languages with subrange type, range checking is done at _____ time.
   a) Compile b) Load c) Run d) Link e) None
5) In C++ declaration "int size", the type "int" is bound to the name "size" at _____ time.
   a) Execution b) Loading c) Compilation d) Language design e) None
6) Implicit declaration is used in a _____ language.
   a) C++ b) Java c) Perl d) Assembly e) None
7) Dynamic type binding is used in a _____ language.
   a) C++ b) Java c) Perl d) Assembly e) None
8) The time period between allocation and de-allocation of a variable is:
   a) Type b) Scope c) Lifetime d) address e) None
9) The variable attribute that specifies where the name is visible in the program is:
   a) Type b) Scope c) Lifetime d) address e) None
10) In C++, the local variables and function parameters are examples of _____ variables.
   a) Stack-dynamic b) Static c) Implicit-heap dynamic d) Explicit-heap

QUESTION TWO: Storage bindings & Lifetime & Array types [10 pts]

Study the following C++ code and answer the questions below:

```
static int x[50];

void moo(int size)
{
  int y[70];
  int w[size];
  int z[]={4,5,6,7,8};
  int *f = new int[120];
  ...
}
char *foo ()
{
  char *pr = new char('R');
  static float d = 2.71;
  return pr;
}
```

- The type of array x is _____.

  a) Static b) Stack-dynamic c) Fixed-stack dynamic d) Fixed-heap dynamic

- The type of array y is _____.

  a) Static b) Stack-dynamic c) Fixed-stack dynamic d) Fixed-heap dynamic

- The type of array w is _____.

  a) Static b) Stack-dynamic c) Fixed-stack dynamic d) Fixed-heap dynamic

- The type of array f is _____.

  a) Static b) Stack-dynamic c) Fixed-stack dynamic d) Fixed-heap dynamic

- The type of a pointer variable pr is **STACK-DYNAMIC**
- The type of the object (variable) pointed to by pr is **EXPLICIT-HEAP DYNAMIC**
- The lifetime of a variable pr begin when **FUNCTION foo IS CALLED** and ends when **FUNCTION foo IS EXITED.**
- The lifetime of a variable d begin when **FUNCTION foo IS CALLED** and ends when **PROGRAM IS TERMINATED**
- The lifetime of array y begin when **FUNCTION moo IS CALLED** and ends when **FUNCTION Moo IS EXITED**
- This function foo suffers from a problem of **MEMORY LEAKAGE**

QUESTION THREE: **Fill in blanks** **[11 pts]**

- The process of binding storage to a name is called **ALLOCATION** the process of breaking the binding between storage and a name is called **DE-ALLOCATION**
- For the purpose of allocating storage to variables, the program storage is divided into 3 parts: Global/Static storage, **STACK** and **HEAP.**
- Pointers are included in languages for two uses: **INDIRECT ADDRESSING** and **DYNAMIC MEMORY MANAGEMENT**
- In non-associative arrays, elements are indexed by **ORDINAL VALUES** In associative arrays, elements are indexed by **KEYS.**
- The dynamic length strings are implemented using **LINKED LISTS** or **ADJACENT MEMORY LOCATIONS.**
- The 2 representation forms of decimal type are: **UNPACKED BCD** and **PACKED BCD.**
- The 2 disadvantages of dynamic type binding are: **LESS RELIABLE** and **HIGH COST**
- The 2 character string design issues: **FIXED / VARIABLE LENGTH** and **PRIMITIVE OR ARRAY OF CHARACTERS.**
- In currently used programming languages character string are defined as **ARRAY OF CHARACTERS** or **PRIMITIVE.**
- The access function for the array F in the following C++ declaration: "`long F[150];`" is

  `F[i]= base + I * 8`
- The access function for the array F in the following C++ declaration: "`int F[80][60];`" is

  `F[i,j]= base + 4 * (I * 60 + J).`

allocation / deallocation

access ~~sun~~ Indirect accessing

~~manage~~ dynamic memory management

base + 8*i

base + 4(I*60 + J)

base + 4 ((I-0)60

0-79 0-59

F[80][60]

F[i][j] = base + 4 ((i-0) * 60 + (j-0))

Student Name: **Student id:** **Serial #:**

QUESTION #	1	2	3	4	5	TOTAL
MAX POINTS	10	12	12	12	4	
POINTS EARNED	6	10.5	11	11.5	2	41

University of Bahrain
College of Information Technology
Department of Computer Science

ITCS332: Concepts of Programming Languages **SECOND TEST** **Date: DEC 25, 2006**

****

QUESTION ONE: **MCQ Questions** **[10 pts]**

1) The activity of ensuring that the operands of an operator are of compatible types is called _____.
a) Type checking b) Type binding c) Operator binding d) Address binding

2) The application of an operator to an operand of an invalid type causes _____ error.
a) Address b) Lexical c) Scope d) Type e) None

3) In languages with subrange type, range checking is done at _____ time.
a) Compile b) Load c) Run d) Link e) None

4) The variable attribute that specifies the number of bytes allocated to a variable is:
a) Type b) Scope c) Lifetime d) address e) None

5) The "float" type is bound to a range of values at _____ time.
a) Run b) Language implementation c) Compile d) Language design

6) The main disadvantage of _____ variables is that they destroy the program modularity.
a) Static b) Stack-dynamic c) Explicit-heap dynamic d) Global e) None

7) Dynamic type binding is used in a _____ language.
a) C++ b) Perl c) FORTRAN d) Assembly e) None

8) The time period between allocation and de-allocation of a variable is:
a) Type b) Scope c) Lifetime d) address e) None

9) The variable attribute that specifies where the name can be accessed in the program is:
a) Type b) Scope c) Lifetime d) address e) None

10) In C++, the local variables and function parameters are examples of _____ variables.
a) Stack-dynamic b) Static c) Implicit-heap dynamic d) Explicit-heap

QUESTION TWO: Storage bindings, Lifetime and Types

[12 pts

Study the following C++ code and answer ALL questions below:

```
static int y[50];

void moo (int len)
{
  int x[800];
  int z[len];
  int w[]={49,59,69,79,89};
  int *f = new int[222];
  ...
}
char *foo ()
{
    char *pr = new char('U');
    static float d = 2.71;
    double aa;
    return pr;
}
```

static	static
stack-dynamic	stack-dynamic
Fixed_heap	explicit-heap
fixed-stack	Implicit-heap

1) The type of array **f** is _____.
   a) Static b) Stack-dynamic c) Fixed-stack dynamic d) Fixed-heap dynamic

2) The type of array **x** is _____.
   a) Static b) Stack-dynamic c) Fixed-stack dynamic d) Fixed-heap dynamic

3) The type of array **y** is _____.
   a) Static b) Stack-dynamic c) Fixed-stack dynamic d) Fixed-heap dynamic

4) The type of array **w** is _____.
   a) Static b) Stack-dynamic c) Fixed-stack dynamic d) Fixed-heap dynamic

5) The scope of a variable **aa** is dynamic The body of (foo) .

6) The lifetime of a variable **aa** begin when Funct. Foo is called
   and ends when Funct. Foo is Exited .

7) The type of a pointer variable **pr** is stack. dynam.

8) The type of the object (variable) pointed to by **pr** is expli, cit heap dynam.

9) The lifetime of a variable **pr** begin when Funct. Foo is called
   and ends when Funct. Foo is exited .

10) The lifetime of a variable **d** begin when Prog is loaded
    and ends when Prog is exited .

11) The lifetime of array **y** begin when Funct. moo is called
    and ends when Funct. Foo is exited .

12) The function **foo** suffers from a problem of memory leaking

QUESTION THREE: **Fill in blanks** **[12 pts]**

1) The access function for the array F in the following C++ declaration: "`long F[300];`" is
`F[n] = base+` i * 8 .

2) The access function for the array F in the following C++ declaration: "`int F[40][90];`" is
`F[k,j] = base+` 4 * (I * 90 + J) .

3) A Type is a collection of data objects and a set of predefined operations. A list of names accessible at a given program point is called a Name Type capability .

4) The type of an object can be determined using 3 ways: using the declaration statement, Assignment statement, and Inferencing Context .

5) The dynamic length strings are implemented using Linked list or Adjacent memory location .

6) The 2 representation forms of decimal type are: Packed BCD and Unpacked BCD.

7) The 2 disadvantages of dynamic type binding are: less reliable and High Cost .

8) The process of binding storage to a name is called allocation; the process of breaking the binding between storage and a name is called deallocation .

9) For the purpose of allocating storage to variables, the program storage is divided into 3 parts: Global/Static storage, Stack, and Heap .

10) Pointers are included in languages for two uses: indirect Addressing and dynamic Memory Management .

11) In non-associative arrays, elements are accessed by Ordinal Value. In associative arrays, elements are accessed by Key .

12) In currently used programming languages character string are defined as Primitive or Array character .

int *p = new int(50)

QUESTION FOUR:

[12 pts]

- Name any 2 string length options used in programming languages and give C++ declarations illustrating each option.

1) Static string length

```
char str[89];
```

2) Limited dynamic string length

```
#define SIZE 89
char str[SIZE];
```

- Name any 2 common programming errors (problems) caused by pointers and give C++ code illustrating each error

1) Dangling pointer

```
int *p = new int(80);
delete p;
*p = 20;
cout << *p;
```

2) Dereferencing uninitialized pointer

```
int *p;
cout << *p;
```

```
int *p = new int(50);
int *r;
p = r;
```

Memory leakage

- **Consider the following pseudo-code and answer the next 2 questions:**

```
void sub1()
  { int a, b;
  ... <------- 1
  }                          /* end of sub1   */
void sub2()
  { int b, c; sub1; }        /* end of sub2*/
void main()
  { int c, d; sub2 (); }     /* end of main*/
```

1) Assuming a **static-scoped** language, the referencing environment at point 1 is:

a, b from sub1

2) Assuming a **dynamic-scoped** language, the referencing environment at point 1 is:

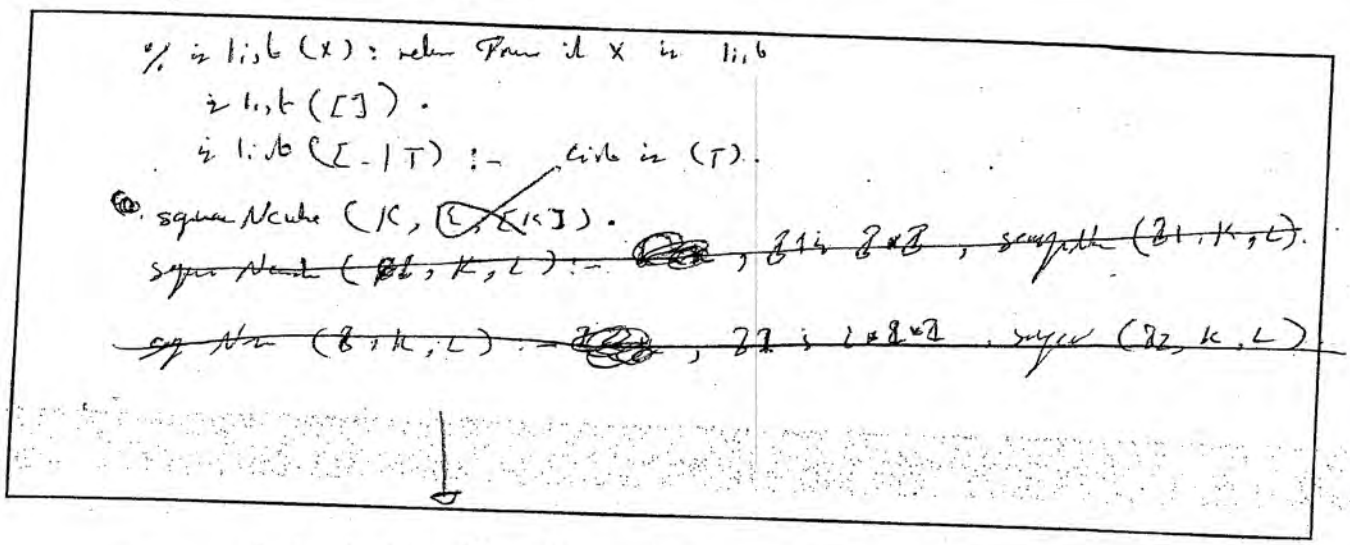
a, b in sub1 c sub2 d main

QUESTION FIVE: Prolog Programming [4 pts]

Write a Prolog predicate(s) named **squareNcube,** that takes a list of values as an argument and returns 2 lists: a list consisting of squares and another list of cubes of all elements in a given list as shown in the following queries.

```
?- squareNcube ([1, 2, 3, 4], M, N).
M = [1, 4, 9, 16]
N = [1, 8, 27, 64]

?- squareNcube ([2, 2.5, 3.0, 4], Squares, Cubes).
Squares = [4, 6.25, 9.0, 16]
Cubes = [8, 15.625, 27.0, 64]
```



```
sq([],M,N) :- M is [], N is [].
sq([H|T],M,N) :- M is [H^^2|T],
```

~~N is [H^^3~~
M is [[H^^2]+H|T].

✓ Append code

Ali

Student Name: **Student id:** **sect#: A** **Serial #:**

QUESTION #	1	2	3	4	TOTAL
MAX POINTS	12	13	15	10	
POINTS EARNED					

University of Bahrain College of Information Technology

Department of Computer Science

ITCS332: Concepts of Programming Languages **SECOND TEST** **Date: JAN 3, 2008**

****

QUESTION ONE: MCQ Questions [12 pts]

1) The code to access any array element must be generated at _____ time.
   a) load b) run c) compile d) language design e) None

2) A ____ is used by compilers for type checking and building the code for allocation /deallocation
   a) data type b) array c) variable d) descriptor e) None

3) Dynamic type binding is used in a _____ language.
   a) C++ b) Perl c) JAVA d) C# e) None

4) The variable attribute defined as a time period between allocation and de-allocation of a variable is:
   a) Compile b) Run c) Lifetime d) Link e) None

5) Dynamic type binding requires type checking at _____ time.
   a) Run b) Load c) Compile d) Link e) None

6) The variable attribute that specifies where the name can be accessed in the program is:
   a) Type b) Scope c) Lifetime d) address e) None

7) In languages with subrange type, type checking is done at _____ time.
   a) Compile b) Load c) Run d) Link e) None

8) A _____ is a collection of memory cells that stores all variable attributes.
   a) Scope b) descriptor c) data type d) Lifetime e) None

9) One of the following is NOT of ordinal type:
   a) bool b) int c) char d) float e) None

10) The subprogram call is bound to subprogram code at _____ time.
   a) Compile b) Language implementation c) Run d) Language design

11) The activity of ensuring that the operands of an operator are of compatible types is called _____.
   a) Type checking b) Type binding c) Operator binding d) Address binding

12) The main disadvantage of _____ variables is that they destroy the program modularity.
   a) Static b) Stack-dynamic c) Explicit-heap dynamic d) Global e) None

Question #	1	2	3	4	5	6	7	8	9	10	11	12
Answer	C	D	B	C	A	B	A	B	D	C	A	D

QUESTION THREE: **Fill in blanks** **[15 pts]**

1) In currently used programming languages character string are defined as **Primary** or **Structured**.

2) Dynamic Scope is based on **CALLING SEQUENCES** of program units, not their spatial textual layout. The variables allocated with `new` and deallocated with `delete` operator are of **EXPLICIT-HEAP DYNAMIC** type.

3) A **DATA TYPE** is a collection of data objects and a set of predefined operations. A list of names accessible at a given program point is called a **REFERENCING ENVIRONMENT.**

4) The 2 disadvantages of dynamic type binding are: **less reliable by making Type error detectionby the compiler is difficult** and **High cost by increasing execution time**

5) Name 2 character string design issues: **Is it a primitive type or just a special kind of array?** and **Should the length of strings be static or dynamic?**

6) The 2 types of type compatibility are: **NAME COMPATIBILITY** and **STRUCTURE COMPATIBILITY**

7) The advantage of implicit declaration is **BETTER WRITEABILITY** and the disadvantage is **LESS READABILITY AND LESS RELIABILITY**

8) Name two problems associated with pointers: **DANGLING POINTERS** and **MEMORY LEAKAGE** and **DEREFERENCING UNINITIALIZED POINTERS**

9) The access function for the array F in the following C++ declaration: "`long F[80];`" is `F[71]= base+` **`(71 - 0) * 8.`**

10) The access function for the array G in the following C++ declaration: "`int G[120][66];`" is `G[50][32]= base+` **`[ (50 - 0)* 66 + (32 - 0) ]* 4.`**

11) Pointers are included in languages for two purposes: **Indirect addressing** and **Dynamic Memory management.**

12) The dynamic length strings are implemented using **Linked Lists** or **Adjacent memory Cells.**

13) The process of binding storage to a name is called **ALLOCATION** ; the process of breaking the binding between storage and a name is called **DE-ALLOCATION.**

14) For the purpose of allocating storage to variables, the program storage is divided into 3 parts: Global/Static storage, **STACK** , and **HEAP** .

15) In non-associative arrays, elements are accessed by **SUBSCRIPTS** . In associative arrays, elements are accessed by **KEYS** .

QUESTION FOUR: Consider the following Ada-like code and answer the next 3 questions **[10 pts]**

```
void main ()
void sub1()
{    int a, b, c;
     ...    < --- 1
}                              /* end of sub1    */
void sub2 ()
{ int b, c, d;
     ...    < ---- 2
     sub1;
}                              /*  end of sub2    */

{ int a, c, d;                 /* begin  of main */
     ...    < ---- 3
     sub2 ( );
}                              /*  end of main    */
```

1) Assuming a static -scoped language, the referencing environment at point 2 is:

2 b, c, d from sub2
a from main

2) Assuming a static -scoped language, the referencing environment at point 1 is:

2 a, b, c from sub1
d from main

3) Assuming a dynamic-scoped language, the referencing environment at point 1 is:

2 a, b, c from sub1
d from sub2

- **Consider the following Ada-like code.** The value of n printed by **print (n) :**

```
Procedure main is
    n: integer ;
Procedure sub1 is
   Begin
   n = n/2;print (n) ;
End ;
Procedure sub2 is
    n : integer;
   Begin
     n = 20  ;  sub1 ;  n = n * 2;
   End ;
   Begin
     n =  36 ; sub2 ;
   End ;
```

2

4) Under static-scoped rules is → 18

5) Under dynamic-scoped rules is → 10

n = 36
n = 20

2 n = 36

10

Student Name: Ahmed Yusuf Jaffar Student id: 20037180 sect#: 1 Serial #: 13

QUESTION #	1	2	3	4	TOTAL
MAX POINTS	15	~~12~~ 11	15	10	42
POINTS EARNED	12	10	10	10	

University of Bahrain College of Information Technology

Department of Computer Science

ITCS332: Concepts of Programming Languages **SECOND TEST** **Date: MAY 21, 08**

****

QUESTION ONE: **[10+5 pts]**

1) `?- [1, 2|X] = [1, 2, [3, 4], 5].`

a) error b) X = [3,4,5] c) X = [3, 4, 5] (d) X = [[3,4],5] e) None

2) `?- Y is X+2, X=1.`

a) Y = 1+2, X = 1 b) Y = 3, X = 1 (c) error message d) Y = X+2, X = 1 e) None

3) For the query: "`[A|B] =[[ dog, mouse], lion, [rabbit, cat]].`", Prolog produces:

A = [[dog, mouse]]

B = [lion, [rabbit, cat]]

4) For the query: "`L9=[L4 | f, d],L3=[5, 6,7],L4=[L3 | k,m,n].`", Prolog produces:

L4 = [5, 6, 7, [K, m, n]]

L9 = [5, 6, 7, K, m, n, f, d]

```
What55 (K,K,[K]).
What55 (I,K,[I|L]) :- I > K, I1 is I - 1, What55 (I1,K,L).
```

5) For the query `?- What55(9,6,U).` Prolog produces L = [9, 8, 7, 6]

```
What88 ([],[]).
What88 ([H|T],[HH|TT]) :- HH is H+3, What88 (T,TT).
```

6) For the query `?- What88([5,12,7],U).` Prolog produces U = [8, 15, 10]

7) Write Prolog predicates named "`listneg`" to count the number of negative values in a given list.

```
listneg ([], 0).
listneg ([H|T], L1) :- H < 0 , L1 is L+1,
                       listneg (T, L1).
```

QUESTION TWO: **[12 pts]**

Study the following C++ code and answer ALL questions below:

```
static int y[50];
void moo (int len)
{  int x[800];
   int z[len];
   int w[]={49,59,69,79,89};
   int *f = new int[320];
   ...
}
char *foo ()
{    char *pr = new char('U');
     static float d = 2.71;
     double aa;
     return pr;
}
```

1) The type of array z is _____.
   a) Static ✓ (b) Stack-dynamic c) Fixed-stack dynamic d) Fixed-heap dynamic

2) The type of array y is _____.
   ✓ (a) Static b) Stack-dynamic c) Fixed-stack dynamic d) Fixed-heap dynamic

3) The type of array f is _____.
   a) Static (b) Stack-dynamic c) Fixed-stack dynamic ✓ d) Fixed-heap dynamic

4) The type of array x is _____.
   a) Static b) Stack-dynamic (c) Fixed-stack dynamic d) Fixed-heap dynamic

5) The type of the object (variable) pointed to by **pr** is explicit heap dynamic

6) The lifetime of a variable **pr** begins when declaration or start of function foo()
   and ends when end of function foo()

7) The lifetime of a variable **d** begins when start of program
   and ends when end of program

8) The function **foo** suffers from a problem of memory leakage

9) The scope of a variable **d** is function foo

10) The scope of a array **x** is function moo

11) The lifetime of a variable **aa** begins when start of function foo()
    and ends when end of function foo()

12) The lifetime of array y begin when start of program
    and ends when end of program

10

QUESTION THREE: ***Fill in blanks Questions*** **[15 pts]**

1) The code to access any array element must be generated at compile time.
Dynamic type binding requires type checking at Run time.

2) The subprogram call is bound to subprogram code at compile Run time.
A descriptor is a collection of memory cells that stores all variable attributes.

3) The variable attribute defined as a time period between allocation and de-allocation of a variable is called lifetime. The variable attribute that specifies where the name can be accessed in a program is called scope.

4) In currently used programming languages character string are defined as
Array of characters or primitive X.

5) Dynamic Scope is based on calling sequence of program units, not their spatial textual layout. The variables allocated with new and deallocated with `delete` operator are of explicit heap dynamic X type.

6) A ~~records~~ data type is a collection of data objects and a set of predefined operations.
A list of names accessible at a given program point is called a ~~local variables~~ referencing environment

7) The 2 disadvantages of dynamic type binding are: High cost
and ~~two parameter variable~~ unreliable

8) Name 2 character string design issues: ~~#define~~ primitive since too Size
and char ~~String~~ [20] Type Fixed ~~length~~ length/variable

9) The 2 types of type compatibility are: name compatibility and Structured Compatibility

10) The advantage of implicit declaration is writability? ? ← Better ~~writability~~
and the disadvantage is unreliable or less readability

11) Given the C++ declaration: "`long F[80];`", the address of the array element `F[30]` is:
base+ 30 * 8

12) Given the C++ declaration: "`int G[120][80];`", the address of the array element `G[75][40]` is
base+ 4 * (75 * 80 + 40) ~~* 80~~

13) The dynamic length strings are implemented using ~~#define~~ or ~~Fixed~~ arrays of strings

14) The process of binding storage to a name is called allocation; the process of breaking the binding between storage and a name is called deallocation

15) For the purpose of allocating storage to variables, the program storage is divided into 3 parts:
Global/Static storage, Stack, and heap.

10 link list & Nested

Memory cell

(80 * 75 + ~~5~~ 40) * 4

QUESTION FOUR: Consider the following Ada-like code and answer the next 3 questions **[10 pts]**

```
void main ()
void sub1()
{    int a, b, c;
     ...    < ---- 1
}                          /* end of sub1     */
void sub2 ()
{ int b, c, d;
     ...  < ----- 2
     sub1;
 }                         /*  end of sub2     */

{ int a, c, d;             /* begin  of main  */
     ... < ----- 3
     sub2 ( );
 }                         /*  end of main     */
```

1) Assuming a **static -scoped** language, the referencing environment at point 2 is:

b, c, d from sub2 & a from main

2) Assuming a **static -scoped language**, the referencing environment at point 1 is:

a, b, c from sub 1 & d from main

3) Assuming a **dynamic-scoped language**, the referencing environment at point 1 is:

a, b, c from sub 1 & d from sub 2

- **Consider the following Ada-like code.** The value of n printed by **print (n) :**

```
Procedure main is
     n: integer ;
Procedure sub1 is
  begin
     n = n * 2 ; print (n) ;
  end ;
Procedure sub2 is
     n : integer;
  begin
     n = 20  ;  sub1 ;  n = n / 2;
  end ;
  begin
     n =  36 ; sub2 ;
  end ;
```

4) Under static-scoped rules is 72 or 36*2

5) Under dynamic-scoped rules is 40

10

Student Name: Ali A. Hassan Emahim Student id: 20036429 sect#: A Serial #:

QUESTION #	1	2	3	4	TOTAL
MAX POINTS	12	15	10	14	40.5
POINTS EARNED	11.5	12	8	9	

University of Bahrain College of Information Technology

Department of Computer Science

ITCS332: Concepts of Programming Languages **SECOND TEST** **Date: AUG 13, 2008**

****

QUESTION ONE: Study the following C++ code and answer ALL questions below: **[12 pts]**

```
static int z[40];
void moo (int len)
{  int w[80];     int y[len];
   int x[]={49,59,69,79,89};
   int *f = new int[32];
   ...
}
Void char *foo ()
{    char *uptr = new char('U');
     static float d = 2.71;
     double sum;    return;
}
```

1) The type of array z is static ✓.
2) The type of array y is stack dynamic ✓.
3) The type of array f is fixed heap dynamic ✓.
4) The type of array w is fixed stack dynamic ✓.
5) The type of the object (variable) pointed to by **uptr** is Explicit heap dynamic.
6) The lifetime of a variable **uptr** begins when function foo is called
   and ends when function foo terminated ✓.
7) The lifetime of a variable **d** begins when ~~function foo is called~~ Begin of program
   and ends when ~~End of program~~ function foo called first time foo is called ✓.
8) The scope of a variable **d** is function foo ✓.
9) The scope of a array **x** is function moo ✓.
10) The lifetime of a variable **sum** begins when function foo is called
    and ends when function foo terminated ✓.
11) The type of a pointer variable **uptr** is stack dynamic ✓.
12) The lifetime of array **z** begin when Begin of program
    and ends when end of program ✓.

11.5

QUESTION TWO: MCQ Questions [15 pts]

1) For every assignment to a subrange variable, types are checked for compatibility at _____ time.
a) Compile b) Load c) Run d) Language design e) Link

2) For every assignment to a subrange variable, range checking is done at _____ time.
a) Compile b) Load c) Run d) Language design e) Link

3) In C++, the arrays allocated/de-allocated using new/delete operators are of type _____
a) Fixed heap dynamic b) Heap dynamic c) Fixed stack dynamic d) None

4) According to IEEE Floating-Point Standard 754, the parts of a real number are arranged:
a) Sign-Fraction-Exp b) Sign-Exp-Fraction c) Exp-Sign- Fraction d) None

5) One of the following is NOT of ordinal type:
a) bool b) int c) char d) double e) None

6) In the statement `f=b+c;` the resulting value is bound to a variable f at _____ time.
a) Compile b) Load c) Run d) Language design e) Link

7) The code to access an array element must be generated by the _____.
a) Loader b) Linker c) Windows d) Compiler e) None

8) One of the following is NOT determined by the data type of an object
a) Operations b) Values Range c) Precision d) Scope

9) A character string whose length is specified when it is declared is called _____ string.
a) Dynamic length b) Static length c) Limited dynamic length d) Fixed length

10) The access function for the C++ declaration: "`int a[][100];`" is `a[i,j]= base+`
a) `(i)*100*4+(j)*4`
b) `(i-1)*100*4-(j-1)*4`
c) `(i-1)*99*4+(j-1)*4`
d) `(i-1)*99*4-(j-1)*4`

11) The array that can grow and shrink during the execution time is of type _____.
a) Fixed heap dynamic b) Heap dynamic c) Fixed stack dynamic d) Static

12) The variable attribute defined as a time period between allocation and de-allocation of a variable is:
a) Type b) Lifetime c) Scope d) Address e) None

13) The variable attribute that specifies where the name can be accessed in the program is:
a) Type b) Scope c) Lifetime d) Address e) None

14) A _____ is a collection of memory cells that stores all variable attributes.
a) Scope b) Data type c) Descriptor d) Lifetime e) None

15) The main disadvantage of _____ variables is that they destroy the program modularity.
a) Global b) Stack-dynamic c) Explicit-heap dynamic d) Static e) None

Question #	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Answer	a	c	a	b	d	c	d	a	b	a	b	b	b	c	a

QUESTION THREE: Consider the following Ada-like code and answer the next 3 questions **[10 pts]**

```
void main ()
void sub1()
{    int a, b, c;
     ...    < --- 1
}                              /* end of sub1    */
void sub2 ()
{ int b, c, d;
     ...    < ---- 2
     sub1;
 }                             /*  end of sub2    */

{ int a, c, d;                 /* begin  of main */
     ...    < ---- 3
     sub2 ( );
 }                             /*  end of main    */
```

1) Assuming a static -scoped language, the referencing environment at point 2 is:

b, c, d from sub2
a of Main

2) Assuming a static -scoped language, the referencing environment at point 1 is:

a, b, c from sub1
d of Main

4

3) Assuming a dynamic-scoped language, the referencing environment at point 1 is:

a, b, c from sub1
d from sub2

- **Consider the following Ada-like code.** The value of n printed by **`print (n)`** :

```
Procedure main is
   n: integer ;
 Procedure sub1 is
   Begin
    n = n/2;print (n) ;
 End ;
 Procedure sub2 is
     n : integer;
   Begin
      n = 10  ;  sub1 ;  n = n * 2;
   End ;
   Begin
      n =  24 ; sub2 ;
   End ;
```

4) Under dynamic-scoped rules is 5

5) Under static-scoped rules is 12

4

spatial textual layout

(5) special Textual layout

QUESTION FOUR: **Fill in blanks** **[14 pts]**

1) Name 2 methods (constructs) used to create aliases in programming languages:
pointers (references) and Sub program.

2) In languages with static type binding, the type may be specified by
Type Checking Explicit declaration and implicit declaration

3) In languages with dynamic type binding, the type may be specified by executing
executing assignment statement and Type inference contents of variable.

4) In currently used programming languages character string are defined as primitives
or array of characters special textual layout.

5) Dynamic scope of variables is based on calling sequence of program units enclosing
those variables. While static scope is based on the scope range of program units

6) The 2 disadvantages of dynamic type binding are: less reliable
and High cost.

7) Name two problems associated with pointers: dangling pointers
and Memory leakage.

8) The access function for the array F in the following C++ declaration: "`long F[80];`" is
`F[71]= base+` 8 * 71

9) The access function for the array G in the following C++ declaration: "`int G[120][66];`" is
`G[50][32]= base+` 4 * (50 * 66 + 32)

10) Pointers are included in languages for two purposes: indirect addressing
and Dynamic memory management.

11) The dynamic length strings are implemented using pointers linked list
or array of characters Adjacent memory cells.

12) The process of binding storage to a name is called allocation; the proces
of breaking the binding between storage and a name is called deallocation

13) For the purpose of allocating storage to variables, the program storage is divided into 3 parts: Global/Static
storage, Stack, and Heap.

14) In non-associative arrays, elements are accessed by Ordinal values subscr
In associative arrays, elements are accessed by keys.

Student Name: Sayed Abbas Hashim **Student id:** 200545538 **Sect#:** 1 **Serial #:** 21

QUESTION #	1	2	3	4	TOTAL
MAX POINTS	12	12	15	15	49
POINTS EARNED	10	11.5	13.5	14	

University of Bahrain — College of Information Technology

Department of Computer Science

ITCS332: Concepts of Programming Languages **SECOND TEST** **Date: DEC 31, 08**

****

QUESTION ONE: **[6+6 pts]**

- Define a prolog predicate `splitn(L,N,L1,L2)` to split a list L into two parts L1 and L2; the length of the first part is N. **For example,**

```
?- splitn([a,b,c,d,e,f,g,h,i,k],3,L1,L2).
L1 = [a,b,c]
L2 = [d,e,f,g,h,i,k].
```

4

splitn(L,0,[],L).

splitn([H|T],N,[H|R],S):- N>0, N1 is N-1, splitn(T,N1,R,S).

- **Consider the following Ada-like code.** The value of n printed by **`print (n)`** :

```
Procedure main is
     n: integer ;
 Procedure sub1 is
   begin
      n = n*2;print (n) ;
   end ;
 Procedure sub2 is
      n : integer;
   begin
      n = 10  ;  sub1 ;  n = n / 2;
   end ;
   begin
      n =  18 ; sub2 ;
   end ;
```

1) Under static-scoped rules is 6 ~~36~~

2) Under dynamic-scoped rules is ~~20~~

QUESTION TWO: Study the following C++ code and answer ALL questions below: **[12 pts]**

```
static int z[40];
void FIFI (int len)
{  int w[80];      int y[len];
   int x[]={49,59,69,79,89};
   int *f = new int[32];
   ...
}
Void char *GIGI ()
{    char *uptr = new char('U');
     static float d = 2.71;
     double sum;   return;
}
```

1) The scope of a variable **d** is function GIGI
2) The scope of a array **x** is function FIFI
3) The lifetime of a variable **sum** begins whenever function GIGI is called
   and ends when function GIGI is termened
4) The type of a pointer variable **uptr** is stack-dynamic
5) The lifetime of array **z** begin when the program is loaded
   and ends when the program is exited
6) The type of the object (variable) pointed to by **uptr** is explict-heap dynamic
7) The type of array f is fixed heap dynamic
8) The type of array w is fixed stack dynamic
9) The lifetime of a variable **uptr** begins whenever function GIGI is called
   and ends when ~~the program is exited~~ the function GIGI is termenated
10) The lifetime of a variable **d** begins when function GIGI is called? loaded
    and ends when the program is exited
11) The type of array z is static
12) The type of array y is stack dynamic

11.5

QUESTION THREE: ***Fill in blanks Questions*** **[15 pts]**

1) Given the C++ declaration: "`int G[120][80];`", the address of the array element `G[75][40]` is base+ 4[(75-0)*80+40]

2) The dynamic length strings are implemented using linked list or Adjacent Memory cells

3) The process of binding storage to a name is called Allocation; the process of breaking the binding between storage and a name is called de-Allocation

4) For the purpose of allocating storage to variables, the program storage is divided into 3 parts: Global/Static storage, heap, and Stack.

5) The code to access any array element must be generated at compile time. Dynamic type binding requires type checking at run time.

6) The subprogram call is bound to subprogram code at run time. A decriptor is a collection of memory cells that stores all variable attributes.

7) The variable attribute defined as a time period between allocation and de-allocation of a variable is called ~~life~~ life time. The variable attribute that specifies where the name can be accessed in a program is called Scope.

8) In currently used programming languages character string are defined as primary or structred.

9) Dynamic Scope is based on calling sequence of program units, not their spatial textual layout. The variables allocated with new and deallocated with `delete` operator are of explict heap dynamic type.

10) A datatype is a collection of data objects and a set of predefined operations. A list of names accessible at a given program point is called a referencing environment

11) The 2 disadvantages of dynamic type binding are: High cost how? and less reliability how-

12) Name 2 character string design issues: primitive or ? and array of characters - fixed length variable

13) The 2 types of type compatibility are: Name compatability and structred compatability

14) The advantage of implicit declaration is better writability and the disadvantage is less readability & less reliability

13.5

15) Given the C++ declaration: "`long F[80];`", the address of the array element `F[30]` is: base+ 8*30

QUESTION FOUR: **MCQ Questions** **[15 pts]**

1) A character string whose length is specified when it is declared is called _____ string.
   a) Dynamic length b) Static length c) Limited dynamic length d) Fixed length

2) The access function for the C++ declaration: "`int a[][100];`" is `a[i,j]= base+`
   a) `(i)*100*4+(j)*4` b) `(i-1)*100*4-(j-1)*4`
   c) `(i-1)*99*4+(j-1)*4` d) `(i-1)*99*4-(j-1)*4`

3) The array that can grow and shrink during the execution time is of type _____.
   a) Fixed heap dynamic b) Heap dynamic c) Fixed stack dynamic d) Static

4) The variable attribute defined as a time period between allocation and de-allocation of a variable is:
   a) Type b) Lifetime c) Scope d) Address e) None

5) The variable attribute that specifies where the name can be accessed in the program is:
   a) Type b) Scope c) Lifetime d) Address e) None

6) A _____ is a collection of memory cells that stores all variable attributes.
   a) Scope b) Data type c) Descriptor d) Lifetime e) None

7) The main disadvantage of _____ variables is that they destroy the program modularity.
   a) Global b) Stack-dynamic c) Explicit-heap dynamic d) Static e) None

8) For every assignment to a subrange variable, types are checked for compatibility at _____ time.
   a) Compile b) Load c) Run d) Language design e) Link

9) For every assignment to a subrange variable, range checking is done at _____ time.
   a) Compile b) Load c) Run d) Language design e) Link

10) In C++, the arrays allocated/de-allocated using new/delete operators are of type _____
   a) Fixed heap dynamic b) Heap dynamic c) Fixed stack dynamic d) None

11) According to IEEE Floating-Point Standard 754, the parts of a real number are arranged:
   a) Sign-Fraction-Exp b) Sign-Exp-Fraction c) Exp-Sign- Fraction d) None

12) One of the following is NOT of ordinal type:
   a) bool b) int c) char d) double e) None

13) In the statement `f=b+c;` the resulting value is bound to a variable f at _____ time.
   a) Compile b) Load c) Run d) Language design e) Link

14) The code to access an array element must be generated by the _____.
   a) Loader b) Linker c) Windows d) Compiler e) None

15) One of the following is NOT determined by the data type of an object
   a) Operations b) Values Range c) Precision d) Scope

Question #	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Answer	b	a	b	b	b	c	a	a	c	a	X	d	c	d	d

b

Student Name: Sayed Baqer Hadi Student id: 20063201 Sect#: 2 Serial #: 11

QUESTION #	1	2	3	4	TOTAL
MAX POINTS	8	12	16	14	
POINTS EARNED	6	11.5	9.5	13	40

University of Bahrain College of Information Technology

Department of Computer Science

ITCS332: Concepts of Programming Languages **SECOND TEST** **Date: DEC 23, 09**

****

QUESTION ONE: **[4+4 pts]**

- Define a prolog predicate `myLast` that finds out the last element in a given list L. **For example,**

```
?- myLast([8,b,5,d,-3,i,k],F).
F = k

?- myLast([1,2,-9,7,[9,-8]],M).
M = [9, -8]
```

continues.

myLast([X],X).

myLast([H|T],X) :- myLast(T,X).

myList([H|T],M) :- myList(T,M).

2

1

- **Consider the following Ada-like code.** The value of n printed by **`print(n)`** :

```
Procedure main is
    n: integer ;
 Procedure sub1 is
   begin
     n = n*2; print (n) ;
  end ; of sub1
 Procedure sub2 is
     n : integer;
  begin
     n = 8 ;  sub1 ;  n = n % 2;
  end ; of sub2
  begin
     n = 12 ; sub2 ;
  end ; of main
```

1) Under static-scoped rules is 24

2) Under dynamic-scoped rules is 16

4

static: 12*2=24

Dynamic: 12 => 8*2=16

QUESTION TWO: Study the following C++ code and answer ALL questions below: **[12 pts]**

```
static int z[16];
void FIFI (int size)
{  int w[24];     int y[size];
   int x[]={9,5,11,-23,17,29};
   int *f = new int[32];
   ...
}
void char *GIGI ()
{    char *uptr = new char('U');
     static float d = 3.75;
     double sum;
}
```

1) The type of the object (variable) pointed to by **uptr** is explicit heap dynamic
2) The lifetime of a variable **uptr** begins whenever GIGI is called
   and ends when GIGI is terminated
3) The lifetime of a variable **d** begins when GIGI is called for the first time
   and ends when the program terminates
4) The scope of a variable **d** is GIGI body
5) The scope of a array **x** is FIFI body
6) The lifetime of a variable **sum** begins whenever GIGI is called
   and ends when GIGI is terminated
7) The type of a pointer variable **uptr** is stack dynamic
8) The lifetime of array **z** begin when ~~the program starts~~ is loaded
   and ends when the program terminate
9) The type of array z is static
10) The type of array y is ~~[illegible]~~ stack dynamic
11) The type of array f is fixed heap dynamic
12) The type of array w is ~~[illegible]~~ fixed stack dynamic

14.5

QUESTION THREE: ***Fill in blanks Questions*** **[16 pts]**

1) The implementation of a floating-point value consists of a sign field and : exponent number
and precision Fraction.

2) The storage size of a given data type is specified by descriptor Type. The order of storing multidimensional array in main memory is specified by descriptor.

3) In languages with static type binding, the type may be specified by declaration implicit declaration and definition explicit declaration

4) Name 2 character string design issues: which encoding to use? (ASCII, ASCIIZ, ..) primitive and coercion to integer? fixed-length variable

5) In languages with dynamic type binding, the type may be specified by assignment/union? assignment statement and interrelation with other operands and operations

6) In currently used programming languages character string are defined as
structure (array of char) or primitive.

7) Dynamic scope of variables is based on the calling sequence of program units enclosing those variables. While static scope is based on the spatial and textual of program units.

8) The 2 disadvantages of dynamic type binding are: high cost to implement
and difficult error tracing for the Compiler

9) Name two problems associated with pointers: memory leakage
and pointer dangling

10) The access function for the array F in the following C++ declaration: "`char F[80];`" is
`F[60]= base+` 60 * 1 X.

11) The access function for the array G in the following C++ declaration: "`int G[120][66];`" is
`G[60][40]= base+` ((60-0) * 66 + (40-0)) * 4

12) Pointers are included in languages for two purposes: in-direct variable access (indirect) indirect address
and dynamic memory management.

13) The dynamic length strings are implemented using: adjacent memory locations adjacent Memory cells
or stack dynamic Link lists.

14) For the purpose of allocating storage to variables, the program storage is divided into 3 parts:
Global/Static storage, stack, and heap.

15) In associative arrays, elements are accessed by subscript keys. In non-associative arrays, elements are accessed by key ordinal value

16) The process of breaking the binding between storage and a name is called deallocation.
The subprogram call is bound to subprogram code at compile time.

9.5

QUESTION FOUR: **MCQ Questions** **[14 pts]**

1) For every assignment to a subrange variable, types are checked for compatibility at _____ time.
a) Compile b) Load c) Run d) Language design e) Link

2) For every assignment to a subrange variable, range checking is done at _____ time.
a) Compile b) Load c) Run d) Language design e) Link

3) In C++, the arrays allocated/de-allocated using new/delete operators are of type _____
a) Fixed heap dynamic b) Heap dynamic c) Fixed stack dynamic d) None

4) One of the following types is NOT ordinal:
a) bool b) enumeration c) char d) double e) int

5) In the statement: `f=b+c;` the symbol + is bound to addition at _____ time.
a) Compile b) Language design c) Run d) Language implementation e) Link

6) The code to access an array element must be generated by the _____.
a) Loader b) Linker c) Windows d) Compiler e) None

7) One of the following is NOT determined by the data type of an object
a) Operations b) Values Range c) Scope d) Precision

8) A character string whose length is specified when it is declared is called _____ string.
a) Dynamic length b) Static length c) Limited dynamic length d) Circular length

9) The access function for the C++ declaration: "`int a[][64];`" is `a[i][j]= base+`
a) `(i)*64*4+(j)*4`
b) `(i-1)*64*4-(j-1)*4`
c) `(i-1)*63*4+(j-1)*4`
d) `(i-1)*63*4-(j-1)*4`

10) The array that can grow and shrink during the execution time is of type _____.
a) Fixed heap dynamic b) Heap dynamic c) Fixed stack dynamic d) Static

11) The variable attribute defined as a time period between allocation and de-allocation of a variable is:
a) Type b) Lifetime c) Scope d) Address e) None

12) The variable attribute that specifies where the variable can be accessed in a program is:
a) Type b) Lifetime c) Scope d) Address e) None

13) A variable _____ is a collection of memory cells that stores all variable attributes.
a) Scope b) Data type c) Lifetime d) Descriptor e) None

14) The main disadvantage of _____ variables is that they destroy the program modularity.
a) Global b) Stack-dynamic c) Explicit-heap dynamic d) Static e) None

13

Question #	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Answer														

Student Name: **Student id:** **Sect#:** **Serial #:**

QUESTION #	1	2	3	4	TOTAL
MAX POINTS	15	15	16	16	
POINTS EARNED					

University of Bahrain College of Information Technology

Department of Computer Science

ITCS332: Concepts of Programming Languages **SECOND TEST** **Date: MAY 26, 2010**

****

QUESTION ONE: Carefully study the following java-like code **[15 pts]**

```
void main() {
    int a, b, c;
    ...
  }
  void fun1() {
    int b, c, d;
    ...
  }
  void fun2() {
    int c, d, e;
    ...
  }
  void fun3() {
    int d, e, f;
    ...
  }
```

Given the following calling sequences and assuming that dynamic scoping is used, what variables are visible during execution of the last function called? Include with each visible variable the name of the function in which it was defined. ***In your answer, do not include hidden variables and variables from main.***

(d) main calls fun1; fun1 calls fun3.

```
d,e,f of fun3
b,c   of fun1
```

(e) main calls fun1; fun1 calls fun2; fun2 calls fun3.

```
d,e,f of fun3
c     of fun2
b     of fun1
```

(f) main calls fun1; fun1 calls fun3; fun3 calls fun2.

```
c,d,e of fun2
f     of fun3
b     of fun1
```

[15 pts]

QUESTION TWO:

Consider a two-dimensional array U of size *r* by *c* allocated at location *k*. Assume that the size of each element is *s* and 1-based index is used. Give the location (or address) of element `U[m][n]` if:

1) The array is stored in the **column-major** order.

```
Loc(U[m][n]) = k + s * [(n-1)*r + (m-0)]
```

2) The array is stored in the **row-major** order.

```
Loc(U[m][n]) = k + s * [(m-1)*c + (n-1)]
```

- Name 3 types of arrays used in C++ and provide one code example to define each type of the arrays.

3) **Static arrays**

```
static int a[100];
```

4) **Fixed-stack dynamic arrays**

```
void f1(... )
{
     int a[100];
}
```

5) **Fixed-heap dynamic arrays**

```
double *p = new double[100];
```

QUESTION THREE: Study the following C++ code and answer ALL questions below: [16 pts]

```
static int z[16];
void SCOOBY (int size)
{  int w[24];      int y[size];
   int x[]={9,5,11,-23,17,29};
   int *f = new int(32);
   ...
}
void char *JERRY ()
{    char *uptr = new char('U');
     static float d = 3.75;
     double sum;
}
```

1) The scope of a variable **d** is THE FUNCTION JERRY
2) The lifetime of a variable **d** begins when THE FUNCTION JERRY IS FIRSTLY CALLED and ends when THE ENTIRE PROGRAM TERMINATES
3) The scope of a pointer variable **f** is THE FUNCTION SCOOBY
4) The type of a pointer variable **uptr** is STACK-DYNAMIC
5) The lifetime of a variable **sum** begins whenever THE FUNCTION JERRY IS CALLED and ends when THE FUNCTION JERRY IS TERMINATED (RETURN).
6) The type of the object (variable) pointed to by **uptr** is EXPLICIT-HEAP DYNAMIC
7) The access function for the array G in the following C++ declaration: "`double G[20][80];`" is `G[5][36]= base+ [ (5 - 0)* 80 + (36 - 0) ]* 8.`
8) The dynamic length strings are implemented using `Linked Lists` or `Adjacent memory Cells.`
9) For the purpose of allocating storage to variables, the program storage is divided into 3 parts: Global/Static storage, `STACK` , and `HEAP`.
10) In languages with static type binding, the type may be specified by `Implicit Declarations` and `Explicit Declarations`
11) Dynamic scope of variables is based on the `Calling sequence` of program units enclosing those variables. While static scope is based on the `spatial textual layout` of program units.
12) Name two problems associated with pointers: `Dangling Pointers` and `Memory Leakage.`
13) Name 2 character string design issues: `String type: primitive or structured` and `String length: static or dynamic.`
14) The storage size of a given data type is specified by `language Implementer.` The order of storing multidimensional array in main memory is specified by `language Implementer.`
15) In languages with dynamic type binding, the type of any program object may be specified by `context inferencing` and `a value is assigned to the object.`
16) In currently used programming languages string are defined as `Primary` or `Structured`